Efficient Computation of Shortest Path-Concavity for 3D Meshes

Henrik Zimmer
zimmer@cs.rwth-aachen.de

Marcel Campen
campen@cs.rwth-aachen.de

Leif Kobbelt
kobbelt@cs.rwth-aachen.de

Computer Graphics Group
RWTH Aachen University, Germany

Abstract

In the context of shape segmentation and retrieval object-wide distributions of measures are needed to accurately evaluate and compare local regions of shapes. Lien et al. [16] proposed two point-wise concavity measures in the context of Approximate Convex Decompositions *of polygons measuring the distance from a point to the polygon's convex hull: an accurate Shortest Path-Concavity* (SPC) *measure and a Straight Line-Concavity* (SLC) *approximation of the same. While both are practicable on 2D shapes, the exponential costs of SPC in 3D makes it inhibitively expensive for a generalization to meshes [14].*

In this paper we propose an efficient and straight forward approximation of the Shortest Path-Concavity measure to 3D meshes. Our approximation is based on discretizing the space between mesh and convex hull, thereby reducing the continuous Shortest Path search to an efficiently solvable graph problem. Our approach works out-of-the-box on complex mesh topologies and requires no complicated handling of genus.

Besides presenting a rigorous evaluation of our method on a variety of input meshes, we also define an SPC-based Shape Descriptor and show its superior retrieval and runtime performance compared with the recently presented results on the Convexity Distribution *by Lian et al. [12].*

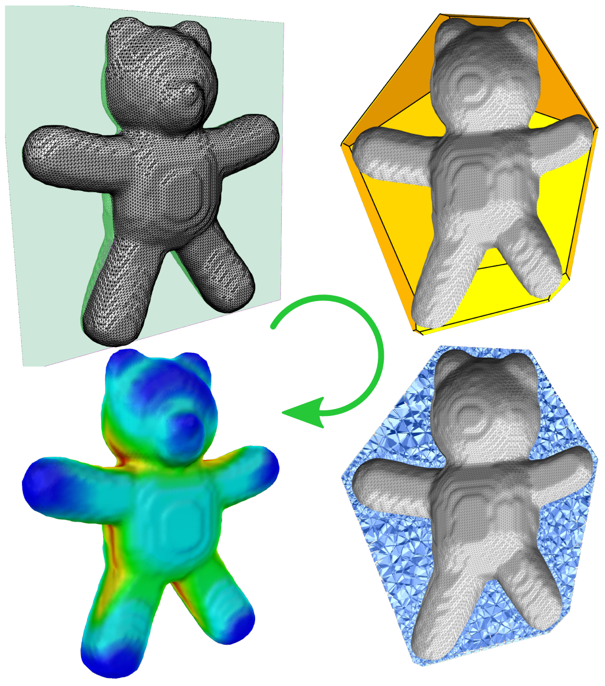


Figure 1. Our algorithm in a nut shell: For an input 3D mesh (top left) a proper outer hull (yellow, top right) is computed to allow for a stable discretization (bottom right) of the free space. By a Fast Marching Method a high accuracy Shortest Path-Concavity measure is obtained for each point on the input. The two right figures are shown in a sliced-through view to visualize the interior.

1. Introduction

Concavity (or convexity) measures for objects find use in different areas of computer vision and geometry processing from shape representation (e.g., [24, 12]) to object decomposition (e.g., [15, 14]).

Concavity (or convexity) is used and interpreted differently in different contexts and does not have a universally accepted definition. Furthermore, while *global convexity measures* define one value per shape, *local convexity measures* yield distributions of values, e.g., on the surface of the shape.

In the context of local convexity an intuitive measure was recently defined based on the shortest path distance from each point on a shape through empty space to its convex hull. This measure is efficiently computable on 2D polygons but an (exact) 3D mesh generalization is prohibitively expensive. In this paper we propose a robust, accurate and efficiently computable approximation of this measure.

1.1. Related Work

Global Convexity Measures. A standard measure for convexity of a 3D (2D) shape $\mathcal{S}$ is the ratio between the volume (area) enclosed by the shape to that of its convex hull $\mathcal{C}_\mathcal{S}$: $C(\mathcal{S}) = \mathrm{Vol}(\mathcal{S}) \cdot \mathrm{Vol}(\mathcal{C}_\mathcal{S})^{-1}$. While this simple measure captures the visual convexity of a shape it has been argued against it as it is insensitive to deep, thin slits in the shape.

A new projection-based global convexity measure, which is able to capture such fine concavities was introduced by Zunic and Rosin for 2D polygons in 2004 [24] and generalized to 3D meshes by Lian et al. in 2012 [12]. It is based on a view-dependent ratio of areas, namely the area of the projected silhouette of a mesh over the sum of all projected face areas. The global measure is taken as the minimum of this ratio over all orthographic views. In 2D a careful analysis yielded a bound on the number of different views needing evaluation to guarantee the minimum. Analyzing the 3D setting is more intricate and [12] propose to use a Genetic Algorithm to find a minimizing view.

Local Convexity Measures. Local convexity measures have been well investigated for 2D shapes. Our research was motivated by the concavity measures defined by Lien et al. [16] in the context of *Approximate Convex Decompositions* (ACD) of polygons. Here the authors use the concept of *bridges* (edges on the convex hull) and *pockets* (concave interior regions of the shape bounded by a corresponding bridge) to define the *Shortest Path-Concavity* (SPC) and its approximation: *Straight Line-Concavity* (SLC). For each pocket point the concavity is defined as the shortest distance (in the respective metric) to the closest point on the associated bridge.

The more accurate SPC measurement can be computed in linear time (in the number of pocket vertices) in 2D, but an efficient generalization to 3D is not known [14]. Not only are the associations between pockets and bridges ambiguous in 3D but also is computing the SPC measure inhibited by exponential running-time. In the context of robot motion planning the NP-hardness of continuous shortest paths was proved by Canny and Reif [5]. Both approximations [6] and exact algorithms [17] have been proposed, however, they have at least superquadratic complexity even for the sub-problem of computing the shortest path between a pair of given points. Thus, in the 3D setting the less accurate SLC is commonly used to measure vertex concavity, while a heuristic is employed to "project" bridge faces onto the surface. This projection is a geometrically delicate procedure requiring extra case-handling for certain surface cavities, and the identification of handle loops and reduction of genus on general polygonal meshes [14].

As an alternative to the problematic bridge/pocket projection Mamou and Ghorbel simply use the straight line distance in normal direction to the convex hull [15]. As can be seen in Figure 2 this is a poor approximation as it depends only on the straight distance along the local normal direction and is insensitive to what lies in between.

In the recent Fast-ACD [7] a localized relative concavity is introduced to guide the decomposition.While increasing the efficiency of the ACD and reducing oversegmentation, the concavity measure still deviates from the intuitive one and has not yet been generalized to objects with genus $\neq 0$.

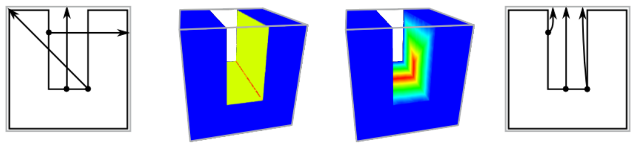


Figure 2. A visual comparison between the SPC (right) and the SLC used in [15] (left) . The SPC accurately computes the shortest distances to the convex hull without self-intersecting the object. The SLC distances measured straight through the object are a poor convexity description depending only the closeness to the convex hull and not what lies in between. Grey lines mark the convex hull.

Convexity as Shape Descriptor. Convexity/Concavity encodes certain characteristics of shapes and can thus be used to derive shape descriptors for 3D retrieval or matching. It should however be noted that, due to their extrinsic nature, convexity/concavity measures are not isometry invariant and hence, at least on their own, not directly suited for dynamic poseable shapes.

Recently Lian et al. carried out 3D retrieval experiments using pure convexity-based and convexity-enriched descriptors [12]. Briefly, their experiments showed two things: that using a distribution of several values is superior to using a single valued descriptor and that retrieval performance can be improved by using composite descriptors additionally enriched by convexity. To obtain a descriptor the authors form a histogram with 1024 bins of their view-dependent *global* convexity measure sampled from 10000 random views. Obtaining these distributions is rather costly, even for small meshes with $5k$ vertices the computation time is in the range of minutes.

Our Contribution. In this paper we present a simple and robust approach to efficiently approximate the Shortest Path-Concavity on 3D meshes[1]. The degree of accuracy can be adjusted to any desired level. Figure 1 visualizes the basic steps of the algorithm. Instead of using visibility trees [8] and bridge-projections as proposed in [16, 14] we propose to tessellate the space between the mesh $\mathcal{M}$ and its convex hull $\mathcal{C}_\mathcal{M}$ and subsequently compute shortest path distances within this discretized space. While a simple graph-based Dijkstra algorithm already yields respectable results, high accuracy is achieved using the Fast Marching Method [19]. Our approach requires no extra identification of handle/tunnel loops and works on meshes of arbitrary genus – they only need to be free of self-intersections.

[1] Source code will be available at: http://www.graphics.rwth-aachen.de/

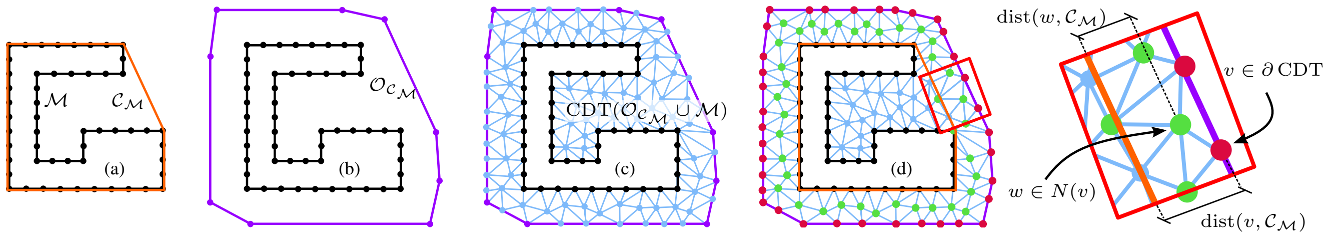

$\mathcal{M}$
$\mathcal{C}_{\mathcal{M}}$
(a)
$\mathcal{O}_{\mathcal{C}_{\mathcal{M}}}$
(b)
$\mathrm{CDT}(\mathcal{O}_{\mathcal{C}_{\mathcal{M}}} \cup \mathcal{M})$
(c)
(d)
$\mathrm{dist}(w, \mathcal{C}_{\mathcal{M}})$
$v \in \partial\,\mathrm{CDT}$
$w \in N(v)$
$\mathrm{dist}(v, \mathcal{C}_{\mathcal{M}})$

Figure 3. Visualization of our approximation algorithm for computing Shortest Path-Concavity. (a) shows the input mesh (black) and the convex hull (orange). In (b) an offset, coarser outer hull has been computed to enable stable Constrained Delaunay Tetrahedralization (CDT). (The offset is exaggerated for visualization purposes.) The CDT of the free space between the input mesh and the outer hull is shown in (c). To compute the shortest distances a Fast Marching Method (FMM) is used. (d) shows the initialization of the FMM: the outer CDT boundary vertices (red) and their inner neighbors (green) are initialized with their distance to the convex hull. Note that while all boundary vertices have a negative distance, their inner neighbors can have positive distances to the convex hull.

We further claim that a per-point concavity measure is more shape descriptive than a (sampled) global measure. In Section 4 we propose a 4-dimensional *scale, tessellation and rotation invariant* descriptor derived from our SPC concavity values, and show how it generally outperforms the 1024-dimensional descriptor of [12] both in retrieval gain and run-time efficiency.

Our experiments demonstrate the efficiency of our approach. Even for models with $260k$ vertices less than 3 minutes are needed to compute the distribution of concavity values, more than one order of magnitude faster than computing the above mentioned descriptor.

This paper is structured as follows: Section 2 describes the proposed SPC algorithm, and Section 3 discusses timings and robustness of the approach. Section 4 presents the SPC-based shape descriptor and shows retrieval results.

2. Efficient Shortest Path-Concavity

For notational convenience we introduce some operators used throughout this paper. For a closed triangle mesh $\mathcal{M}$ embedded in $\mathbb{R}^3$, let $\mathcal{C}_{\mathcal{M}}$ denote the convex hull of, $\mathrm{CDT}(\mathcal{M})$ the Constrained Delaunay Tetrahedralization [21] of, $\mathrm{Vol}(\mathcal{M})$ the volume enclosed by, and $\mathrm{avgel}(\mathcal{M})$ the average edge length of the mesh $\mathcal{M}$.

The goal of the algorithm is to compute a scalar valued concavity measure $C(v) \in [0, \infty)$ for every vertex $v \in \mathcal{M}$. The measure should well approximate the shortest path from v through the volume $\mathrm{Vol}(\mathcal{C}_{\mathcal{M}}) \setminus \mathrm{Vol}(\mathcal{M})$ from the mesh to the convex hull. From a practical point of view the algorithm should additionally be easy to implement.

Algorithm Overview. The algorithm first computes $\mathcal{C}_{\mathcal{M}}$, then tessellates the enclosed space by applying a Constrained Delaunay Tetrahedralization followed by a Fast Marching to get the shortest distances from $\mathcal{M}$ to $\mathcal{C}_{\mathcal{M}}$. The first step is simply solved by applying any convex hull algorithm. The result is a mesh $\mathcal{C}_{\mathcal{M}}$ with $\mathcal{C}_{\mathcal{M}} \cap \mathcal{M} \neq \emptyset$. Unfortunately, the requirement on the constraining surfaces to be free of intersection posed by common CDT algorithms rules out a direct tessellation of $\mathrm{Vol}(\mathcal{C}_{\mathcal{M}}) \setminus \mathrm{Vol}(\mathcal{M})$ due to the non-empty intersection. Hence, for the second step, instead of using $\mathcal{C}_{\mathcal{M}}$ we compute a coarse but tight proper hull (consisting of well shaped elements at an offset from $\mathcal{C}_{\mathcal{M}}$) and adjust the computation of the distances accordingly. The second step of the algorithm is explained in more detail in Section 2.1 below. Figure 3 shows an overview.

2.1. Robust CDT of the Empty Space

Even when employing state of the art meshing libraries (we use `TetGen` [21]) the nature of the Constrained Delaunay Tetrahedralization can still cause problems in practice. The CDT of a mesh $\mathcal{M}$ (the constraining surface) interpolates the elements of $\mathcal{M}$. In particular also very skinny *needle* and *cap* triangles are interpolated. Besides the possible stability issues caused by completely degenerate faces, this results in either poor quality long/skinny tetrahedra or, when constraining element quality, to an overly fine tessellation with very many small elements needed to cover the narrow angled triangle corners. Figure 4 (top row) visualizes the problem. Additionally, when the constraining surface(s) (self-)intersect the inside/outside property of the volume to be tessellated is no longer uniquely defined and the mesher cannot proceed.

While intersections between constraining surfaces intuitively could be handled by identifying and removing (stitching together) touching regions, this is a very delicate procedure geometrically, which still does not fix the problem of "bad" convex hull elements. A (slightly) offset bounding box could be used as an outer constraining surface instead of the convex hull itself, but this would have a negative impact on performance, as generally unnecessarily much free space would have to be tessellated between the bounding box and the mesh.

We overcome these difficulties and obtain a proper and tight outer mesh by simple geometric operations.

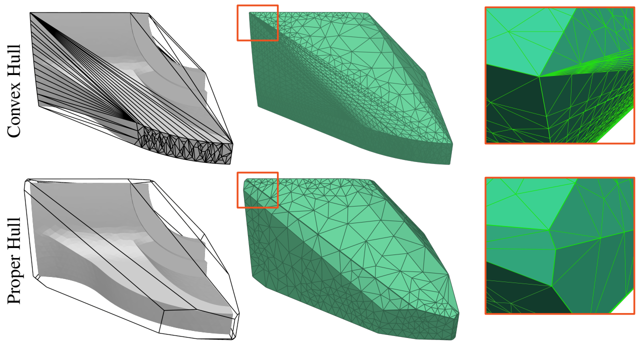


Figure 4. Computing a Constrained Delaunay Tetrahedralization of the convex hull directly (top row) can be unstable and leads to an overly fine tessellation. Using a coarse but tight approximation of the convex hull avoids these problems (cf. Section 2.1)

2.1.1 A Proper Outer Hull

To stably compute a CDT of the free space, a proper mesh $\mathcal{O}_{\mathcal{C}_\mathcal{M}}$ is needed as an outer constraining surface, having the following properties: it should lie completely outside (and not intersect) $\mathcal{C}_\mathcal{M}$ to allow for a stable tessellation, while at the same time lying close to $\mathcal{C}_\mathcal{M}$ and having large well-shaped faces to avoid an overly fine tessellation.

Our proposed outer hull $\mathcal{O}_{\mathcal{C}_\mathcal{M}}$ is in spirit similar to the Discrete Oriented Polytopes (k-DOP) typically used in ray-tracing and collision detection (cf. [9, 23]). However, while in a ray-tracing setting a *fixed* set of planes simplifies intersection tests, we can use a set of *variable* planes derived from the convex hull for a tighter fit and obtain a more object specific outer hull. For this, first the face normals of $\mathcal{C}_\mathcal{M}$ are grouped into k clusters by weighted k-means clustering on the unit sphere. Each normal is weighted by the corresponding face area to account for non-uniform tessellation. The k cluster centers then define k planes which we position to have a certain offset distance $dist$ to their closest point on $\mathcal{C}_\mathcal{M}$.

Implementation Details. Computing proper spherical weighted averages for the clustering can be done using the method by Buss and Fillmore [4]. However, for our scenario a weighted Euclidean averaging followed by a re-projection to the sphere proved sufficient.

To obtain $\mathcal{O}_{\mathcal{C}_\mathcal{M}}$ the intersections of the halfspaces defined by the planes need to be determined. While this could be implemented using plane intersections, the problem can be solved much more efficiently by utilizing the duality between convex hulls and halfspace intersections. We transform each of the k planes $\mathbf{n}_k^\top \mathbf{x} = d_k$ to a normalized representation $\frac{1}{d_k}\mathbf{n}_k^\top \mathbf{x} = 1$ and take $\frac{1}{d_k}\mathbf{n}$ to be a point in dual space (cf. [3]). The center of gravity of the convex hull is used as origin and d_k is the distance of plane k to this point. Now the convex hull of the dual points is computed, and the plane equations of the faces normalized to get points in our primal space again. Note that these points lie about the global origin and need to be translated to the center of gravity of $\mathcal{C}_\mathcal{M}$. The face connectivity (the n incident vertices of a primal face) is directly defined by the dual vertex connectivity (the n faces incident to a dual vertex).

Using a small k we obtain a coarse, but still tight $\mathcal{O}_{\mathcal{C}_\mathcal{M}}$ around $\mathcal{C}_\mathcal{M}$. In practice $k \approx 25$ proved to provide a good balance. For simplicity we initialize the k-means clustering with the $k = 26$ directions corresponding to the corners, edge, and face centers of the unit cube. The offset distance is chosen as $dist = \text{avgel}(\mathcal{M})$. Figure 4 (bottom row) shows an example of such an outer hull and compares the CDT of this hull with that of the convex hull.

Now the space $\text{Vol}(\mathcal{O}_{\mathcal{C}_\mathcal{M}}) \setminus \text{Vol}(\mathcal{M})$ between the coarse outer hull and the input mesh can be robustly tetrahedralized as $\text{CDT}(\mathcal{O}_{\mathcal{C}_\mathcal{M}} \cup \mathcal{M})$.

2.2. Shortest Paths between $\mathcal{M}$ and $\mathcal{C}_\mathcal{M}$

The CDT computed in the previous step tessellates the space $\text{Vol}(\mathcal{O}_{\mathcal{C}_\mathcal{M}}) \setminus \text{Vol}(\mathcal{M})$ with tetrahedra. We could now propagate distance fields of each vertex $v \in \mathcal{M}$ in this space until hitting $\mathcal{C}_\mathcal{M}$ to obtain $C(v)$. It is however much more efficient to compute a single distance field f from $\mathcal{C}_\mathcal{M}$ and then set $C(v) := f(v)$. The problem is that $\mathcal{C}_\mathcal{M}$ is not explicitly represented (as vertices, edges or faces) in the CDT. Hence, we compute the distance field from $\mathcal{O}_{\mathcal{C}_\mathcal{M}}$, initialized with the appropriate (negative) distances to $\mathcal{C}_\mathcal{M}$. This means that the zero iso-surface comes to lie (approx.) at $\mathcal{C}_\mathcal{M}$ as if $\mathcal{C}_\mathcal{M}$ itself had been used as source.

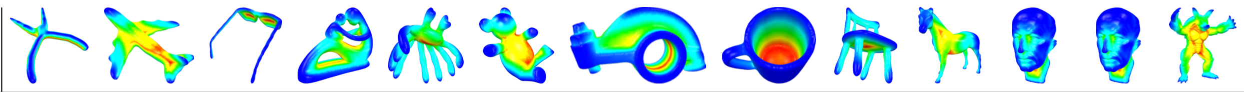
Implementation Details. To initialize the Fast Marching, for the outer boundary vertices $v \in \partial\,\text{CDT}$ we set $\text{tag}(v) = DONE$ and initialize them with their closest (negative) distance to the convex hull: $dist(v) = dist(v, \mathcal{C}_\mathcal{M})$. For their direct non-boundary neighbors $w \in N(v), w \notin \partial\,\text{CDT}$ we set $\text{tag}(w) = CLOSE$ and also initialize them with their signed distance to the convex hull: $dist(w) = dist(w, \mathcal{C}_\mathcal{M})$. All other vertices are tagged FAR and their distances initialized with ∞. A priority queue is set up with the $CLOSE$ vertices sorted in order of ascending distance. Until the queue is empty, vertices are popped from the queue and the distances of their neighbors are updated; FAR neighbors are tagged $CLOSE$ and added to the queue.

3. SPC Results

3.1. Efficiency

Table 1 shows results of our algorithm computed on a variety of meshes. Even for a large mesh with about $260k$ vertices the computation takes less than 3 minutes in total.

The most time is spent computing the Constrained Delanunay Tetrahedralization. t_{CDT} mainly depends on two

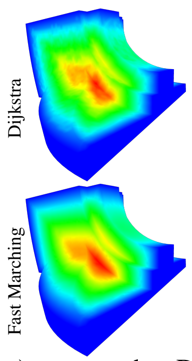


$\lvert \mathcal{M} \rvert$	$11k$	$12k$	$13k$	$14k$	$18k$	$18k$	$21k$	$21k$	$27k$	$35k$	$50k$	$51k$	$260k$
t_{CH}	$0.57s$	$0.6s$	$0.6s$	$0.64s$	$0.9s$	$0.9s$	$1.0s$	$1.0s$	$1.4s$	$1.8s$	$2.0s$	$2.5s$	$13s$
$\lvert \mathcal{C}_{\mathcal{M}} \rvert$	$0.8k$	$0.5k$	$1.7k$	$1.6k$	$1.5k$	$2.7k$	$1.5k$	$4.1k$	$0.9k$	$1.9k$	$6.9k$	$5.9k$	$2.2k$
t_{OH}	0.09	$0.04s$	$0.11s$	$0.12s$	$0.09s$	$0.12s$	$0.10s$	$0.26s$	$0.07s$	$0.11s$	$0.39s$	$0.30s$	$0.11s$
t_{CDT}	$2.7s$	$2.6s$	$3.9s$	$2.7s$	$4.2s$	$3.9s$	$4.4s$	$4.0s$	$7.1s$	$9.74s$	$9.4s$	$46.1s$	$88s$
$\lvert \text{CDT} \rvert$	$17k$	$16k$	$25k$	$18k$	$26k$	$24k$	$27k$	$27k$	$41k$	$52k$	$56k$	$178k$	$524k$
t_{FMM}	$0.51s$	$0.47s$	$0.97s$	$0.52s$	$0.96s$	$0.71s$	$0.76s$	$0.72s$	$1.66s$	$2.12s$	$1.6s$	$8.3s$	$32.4s$
μ	0.0185	0.0390	0.0293	0.0307	0.0418	0.0395	0.0318	0.0958	0.0556	0.0453	0.0121	0.0122	0.0559
σ	0.0186	0.0315	0.0404	0.0357	0.0418	0.0430	0.3215	0.1347	0.0470	0.0422	0.0165	0.0165	0.0470
m_{skew}	0.8692	0.5909	1.6487	1.0637	1.4038	0.9700	1.0066	1.1881	1.1129	0.7735	1.7075	1.6994	0.6914
m_{kurt}	2.5523	2.2539	4.8814	3.0899	5.0331	2.7035	3.2492	2.9053	3.9109	2.4362	5.6484	5.6114	2.3987

Table 1. Results of the SPC computation. The cardinalities $|\mathcal{X}|$ always refer to the number of vertices in the respective mesh. The t_{X} show the timings of the convex hull (CH), the outer hull (OH), the CDT, and the cost of the Fast Marching Method (FMM). The last four rows show the entries of our SPC based shape descriptor (cf. Section 4).

things: the complexity of the input mesh, as all input elements are interpolated, and also on the quality of the input elements. The second and third column (from the right) of Table 1 demonstrate the dependency on the input element quality. The CDT time of the left Max Planck head (an irregular triangle mesh) is almost five times that of the right head (a regular remeshing [2] of the same), both meshes have approximately the same number of elements.

3.2. Accuracy



The Tetrahedral Fast Marching Method (FMM) is well explained in [11]. Implementing FMM is similar to Dijkstra but offers the advantage of first order accuracy, meaning that (1) we can use coarse tessellations and still yield good results, and (2) we can increase accuracy by using a finer tetrahedralization (due to the convergence properties of FMM). The inset figure shows the smoother distribution of values computed by Fast Marching (bottom) compared to Dijkstra (top) on the same tessellation. Figure 5 (left) further shows that quite low SPC errors can be achieved already with rather coarse meshes (compared to a very fine ground truth computation).

Note also that the shape, size and tightness of the proper outer hull (cf. Section 2.1.1) does mainly affect computation times, and not so much the accuracy of the resulting SPC (and hence also the SPCD) values, as the FMM initialization is done w.r.t. the actual convex hull, not the outer hull.

3.3. Robustness

The foremost limitation of our algorithm is the inability to handle intersecting and touching (co-planar) faces of the input. This limitation comes from the employed CDT algorithm [21]. However, even though the CDT algorithm cannot deal with perfectly co-planar faces, Figure 6 (a) shows how our method can operate on visually co-planar, very narrow slits without problems. Additionally, Figure 6 shows that it is robust against irregular triangulations such as the one on the head of Max Plank and the kind of small scale topological noise which exists in some models.

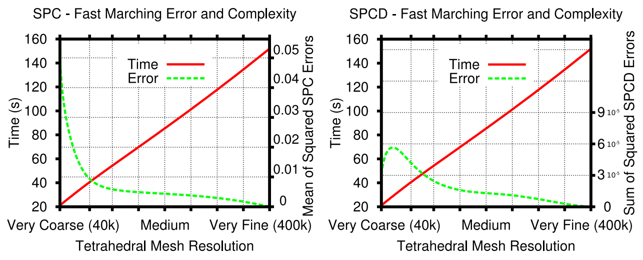


Figure 5. The plots show the effect of increasing the resolution of the CDT mesh on the accuracy of the SPC and that of the SPC-based descriptor (cf. Section 4). After a certain point the already low error decreases only slowly. Enabling major time savings when absolute accuracy is not required.

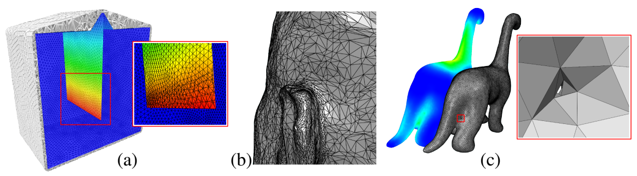
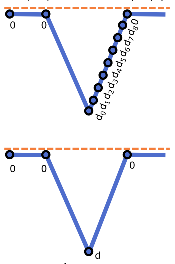


Figure 6. The SPC algorithm can robustly be computed on a box with a thin narrow slit with only 1° opening-angle. (a) a cut through the CDT mesh shows the slit inside, in the closeup the z-fighting artifacts can be seen. (b) a close-up of the irregular triangulation of the Max Plank head (cf. also Table 1). (c) small topological holes without self-intersections are also not a problem.

4. SPC-based Shape Retrieval

Based on the computed convexity values $C(v)$ for each vertex $v \in \mathcal{M}$ we now define a 4-dimensional shape descriptor $SPCD \in \mathbb{R}^4$ of $\mathcal{M}$ based on central moments. More specifically, the entries are the mean, standard deviation, skewness and kurtosis of the concavity values distributed over $\mathcal{M}$: $SPCD = [\mu, \sigma, m_{\text{skew}}, m_{\text{kurt}}]$.

While *rotation* invariance follows directly from the computation of SPC, more care must be taken regarding *tessellation* and *scale*. For the latter we normalize all $C(v)$ by dividing by a scale dependent value of the mesh. Since common choices of such normalizations, e.g., the bounding box diagonal, depend on the orientation of the object, we instead use the square root of the surface area $s = (\int_{\mathcal{M}} dx)^{\frac{1}{2}}$ as normalization, i.e., $C^0(x) = C(x)/s$. Finally, for *tessellation* invariance it is important that the measure only depends on the actual shape of the object and not the distribution or number of vertices. The inset figure shows two different tessellations of the same shape where naïve sampled-based measures would yield different results. To achieve tessellation independence the moments are computed by integration over the piecewise linear surface $\mathcal{M}$, by means of some curvature formula per triangle:



$$\mu = \frac{\int_{\mathcal{M}} C^0(x)dx}{\int_{\mathcal{M}} dx} \qquad \sigma^2 = \frac{\int_{\mathcal{M}} C^0(x)^2 dx}{\int_{\mathcal{M}} dx} - \mu^2$$

$$m_{\text{skew}} = \frac{\int_{\mathcal{M}} (C^0(x) - \mu)^3 dx}{\int_{\mathcal{M}} dx} \frac{1}{\sigma^3}$$

$$m_{\text{kur}} = \frac{\int_{\mathcal{M}} (C^0(x) - \mu)^4 dx}{\int_{\mathcal{M}} dx} \frac{1}{\sigma^4}.$$

In our experiments the trapezoidal rule proved sufficient for this purpose. Refer to Table 1 for examples of these measures for different shapes. Figure 8 furthermore visualizes the (dis-) similarities between different object classes in form of a confusion matrix. At the retrieval stage the dimensions of the SPCD are mean centered and normalized to a standard deviation of one over all classes (z-score) to account for the different value ranges. As distance function we use a weighted L_2 norm with a convex combination of empirically determined weights:

$$[w_\mu, w_\sigma, w_{m_{\text{skew}}}, w_{m_{\text{kurt}}}] = [0.41, 0.34, 0.07, 0.17].$$

4.1. Comparison to State of the Art

We compare the retrieval performance of our SPCD to the CD descriptor proposed by Lian et al. [12]. There, the L_1 norm of the histogram difference is used to compare two descriptors. For evaluation we use the shapes in the McGill 3D Shape Benchmark [22] and compare four quantitative statistical measures (NN, 1-Tier, 2-Tier, DCG, cf. [20]) as well as graphical comparisons in the form of precision-recall curves and confusion matrices. The McGill Benchmark consists of 457 meshes divided into 19 classes, the first 10 consisting of articulated objects (e.g., ants and hands) the last 9 of moderately articulated objects (e.g., tables and airplanes). Lian et al. evaluated their CD shape descriptor on computed canonical forms [13] (non-rigid deformations) of the first 10 classes. We use our reference implementation of [12] to compare the descriptors on these canonical forms and also on the whole McGill dataset.

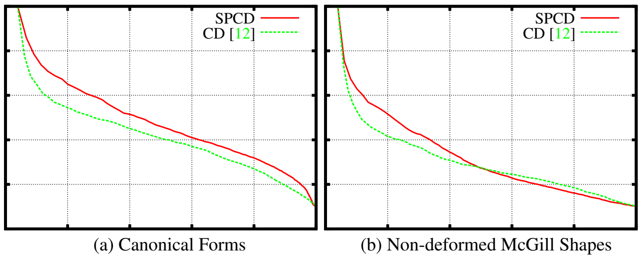


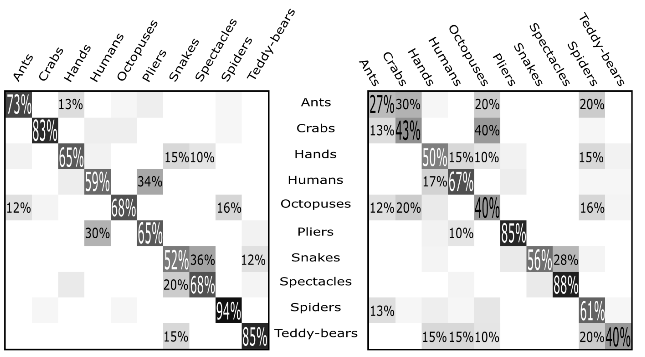
Figure 7. Average Precision-Recall curves on the canonical forms (a) and the non-deformed (b) shapes of the first 10 classes of the McGill Shape Benchmark. The average precision (y-axis) is higher on the less diverse classes of the canonical forms than on the non-deformed McGill shapes. The precision of SPCD (red) is higher on both data sets at low recall rates.

Retrieval Comparison. Table 2 shows how our 4-dimensional SPCD descriptor derived from point-wise concavity measures outperforms the 1024-dimensional CD on the canonical forms of the articulated classes. We achieve a Nearest Neighbor-ratio of 71.7% compared to the 55.2% of [12]. Note that we used remeshed versions of the data sets to be free of self-intersections. Hence the slightly different values (mid row) of our reference implementation compared to the original results in [12]. The precision-recall curves in Figure 7 (a) confirm these results.

Descriptor	NN	1-Tier	2-Tier	DCG
SPCD (Our)	**71.7%**	**44.1%**	**68.7%**	**81.4%**
CD (Rem)	55.2%	40.5%	65.4%	77.5%
CD ([12])	57.3%	41.3%	67.1%	72.9%

Table 2. Retrieval performance comparison between SPCD and CD on the canonical forms of the first 10 classes of the McGill Shape Benchmark. The top and mid row use the same (remeshed) models. The last row states the original values of [12].

For further analysis of the difference between the two descriptors we compare the confusion matrices (cf. Figure 8). The confusion matrices visualize the mis-matches between the classes and reveal classes problematic for both methods. In particular the octopus class, which is visually



(a) Shortest Path-Concavity Descriptor (SPCD) (b) Convexity Distribution (CD) [12]

Figure 8. Confusion matrices on the canonical forms of the first 10 classes of the McGill Shape Benchmark – showing for a query class (row) the percentage of hits in the different classes (columns) for the proposed SPCD (a) and the CD by Lian et al. (b). The SPCD produces less spread, with all diagonal entries above 50%. The rows are normalized to 100% and for a clarity of visualization only entries $\geq 10\%$ are shown.

similar to ants, crabs and spiders, is not well handled by either method. By leaving out the octopus class from the comparison the NN-ratios of both SPCD and CD improve: to 73.4% and 63.0% respectively.

We also performed four retrieval tests of both descriptors against different subsets of the non-deformed models of the McGill Shape Benchmark: on all 19 classes, on the subset of articulated classes, on the moderately articulated classes and on a set of selected classes. Table 3 shows the results. Here the SPCD performs better in all four tests, being significantly better in on the moderately articulated shapes. On the articulated models the SPCD is still better than the CD, however, the result is much worse than on the canonical forms (cf. Table 2). The reason for this is twofold.

Classes		NN	1-Tier	2-Tier	DCG
All	**SPCD**	**51.2%**	**26.6%**	**40.0%**	**68.2%**
	CD [12]	45.9%	24.9%	38.5%	66.9%
Articulated (cf. Table 2)	**SPCD**	**50.9%**	**33.9%**	**50.4%**	**73.6%**
	CD [12]	47.0%	30.8%	50.4%	71.6%
Moderate	**SPCD**	**70.2%**	**37.2%**	**55.7%**	**76.5%**
	CD [12]	62.3%	34.4%	52.1%	74.8%
Selected	**SPCD**	**81.2%**	**52.0%**	**70.5%**	**84.8%**
	CD [12]	79.1%	46.9%	67.7%	83.0%

Table 3. Retrieval performance comparison of SPCD and CD on different subsets of the McGill Shape Benchmark.

Taking a closer look at the canonical forms (cf. Figure 9) reveals that this deformation produces a quite uniform spread of limbs compared to the more diverse poses of the original data set. By reducing the pose-variance the class members are more similar and can hence be more easily matched. This can also be seen on the precision-recall curves in Figure 7 where precision is lower on the non-deformed McGill shapes. Furthermore, descriptors only utilizing convexity (local or global) will never be able to perfectly separate similar classes such as birds from airplanes or spiders from ants (cf. Figure 10). This explains the improved performance in the last row when only considering a selected set of six diverse classes: humans, glasses, airplanes, mugs, dolphins and tables.

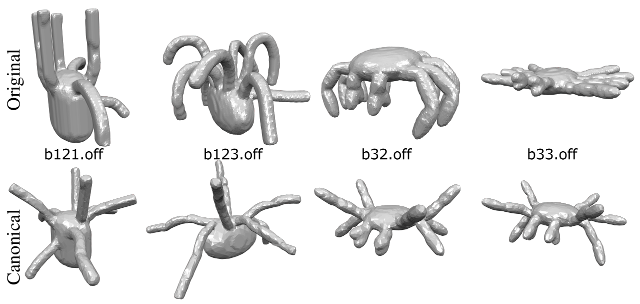


Figure 9. The original McGill Shape Benchmark contains more diverse poses within the classes, leading to a worse retrieval result than on the set of canonical forms of the same.

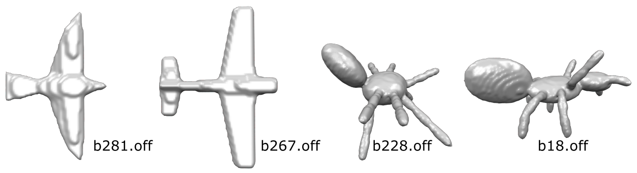


Figure 10. The McGill Shape Benchmark contains classes with very similar objects: e.g., birds, airplanes, spiders and ants.

Efficiency. Computing the 10000 samples for the CD takes a couple of minutes even for relatively small meshes with $5k$ vertices. This is about the same time needed to compute SPC on a mesh with $260k$ vertices (cf. Table 1). Once SPC has been computed, evaluating our SPCD is instantaneous as it requires only a single iteration over the faces of the $\mathcal{M}$ to perform the integration. Also the used normalization – the square root of the surface area – is efficiently computed in the same iteration.

Applicability. Both methods were evaluated on watertight (closed) meshes, in order to guarantee a consistent and unique concavity measure. Both descriptors can also be considered invariant to scalings and rotations. However, while both methods are also robust against different tessellations, our approach is limited by the sensitivity of the used CDT-algorithm. This poses the additional constraint that the meshes be free of self-intersections.

Identifying and cleaning general self-intersections, holes and other inconsistencies is a complex and ambiguous procedure [1], however, all intersections we encountered in our experiments occurred on a very local scale and already a simple remesher such as [2] got rid off these artifacts.

Accuracy. The FMM provides a very accurate approximation - using simple Dijkstra-like shortest paths, possibly even in simpler regular grids, would be possible but would introduce larger errors and a stronger tessellation dependency. While this can be less critical if the SPC values are used in an integral manner for building descriptors, it can strongly disturb scenarios where the SPC field itself is used directly, e.g. for object decomposition or segmentation.

5. Conclusion

We presented a simple and efficient method for computing point-wise Shortest Path-Concavity values on 3D meshes. A distribution of such values cannot only be integral part of segmentation methods, but we further show that it can be used in shape retrieval, where it significantly improves on the results presented recently in [12]. Tested on the full McGill Shape Benchmark the SPCD always demonstrated a NN ratio greater than 50% and turned out to be especially effective on moderately articulated shapes.

The efficiency of our method is evaluated on a variety of different meshes and using a Fast Marching approach we compute accurate distances already at low CDT resolutions. However, when applied to high resolution, irregular input meshes the CDT time will eventually become a bottle-neck. The two Max Planck heads in Table 1 demonstrate exemplary how the CDT timings can be reduced siginificantly by uniformly remeshing the input, without visually altering the SPC or significantly changing the descriptor values.

Acknowledgements

We are grateful to Anca Ivanescu and Torsten Sattler for the valuable discussions, Mike Kremer for help with the `OpenVolumeMesh` data structure [10], and Jan Möbius for support on `OpenFlipper` [18]. This work was funded by the DFG Cluster of Excellence UMIC (DFG EXC 89).

References

[1] M. Attene, M. Campen, and L. Kobbelt. Polygon mesh repairing: An application perspective. *ACM Comput. Surv.*, 45(2):15:1–15:33, 2013. 7

[2] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *SGP*, pages 185–192, 2004. 5, 8

[3] K. Brown. *Fast Intersection of Half Spaces*. Defense Technical Information Center, 1978. 4

[4] S. R. Buss and J. P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM TOG*, 20(2):95–126, Apr. 2001. 4

[5] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Symp. Found. of Comp. Sci.*, pages 49 –60, oct. 1987. 2

[6] K. Clarkson. Approximation algorithms for shortest path motion planning. In *STOC*, pages 56–65, 1987. 2

[7] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494 – 504, 2013. 2

[8] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. In *SoCG*, pages 1–13, 1986. 2

[9] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. SIGGRAPH, pages 269–278, 1986. 4

[10] M. Kremer, D. Bommes, and L. Kobbelt. OpenVolumeMesh - a versatile index-based data structure for 3d polytopal complexes. In X. Jiao and J.-C. Weill, editors, *Proc. Int. Meshing Roundtable*, pages 531–548, 2012. 8

[11] P. G. Lelivre, C. G. Farquharson, and C. A. Hurich. Computing first-arrival seismic traveltimes on unstructured 3-d tetrahedral grids using the fast marching method. *Geophysical Journal International*, 184(2):885–896, 2011. 5

[12] Z. Lian, A. Godil, P. Rosin, and X. Sun. A new convexity measurement for 3d meshes. In *IEEE CVPR*, pages 119–126, 2012. 1, 2, 3, 6, 7, 8

[13] Z. Lian, A. Godil, and J. Xiao. Feature-preserved 3d canonical form. *IJCV*, pages 1–18, 2012. 6

[14] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. In *SPM' 07*, pages 121–131. 1, 2

[15] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *IEEE ICIP*, pages 3501 –3504, nov. 2009. 1, 2

[16] J. ming Lien and N. M. Amato. Approximate convex decomposition of polygons. In *SoCG '04*, pages 17–26. 1, 2

[17] J. S. B. Mitchell and M. Sharir. New results on shortest paths in three dimensions. In *SoCG*, pages 124–133, 2004. 2

[18] J. Möbius and L. Kobbelt. OpenFlipper: An open source geometry processing and rendering framework. In Boissonnat et al., editor, *Curves and Surfaces*, volume 6920 of *LNCS*, pages 488–500. Springer Berlin / Heidelberg, 2012. 8

[19] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999. 2

[20] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *SMI '04*, pages 167–178. 6

[21] H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proc. Int. Meshing Roundtable*, pages 147–163, 2005. 3, 5

[22] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces. *MVA*, 19(4):261–275, 2008. 6

[23] G. Zachmann. Rapid collision detection by dynamically aligned dop-trees. In *VRAIS*, pages 90–97, 1998. 4

[24] J. Zunic and P. Rosin. A new convexity measure for polygons. *IEEE PAMI*, 26(7):923 –934, july 2004. 1, 2